Exhibit II

```
#!/usr/local/bin/perl
# COPYRIGHT  (c) 1998 Telcordia Technologies Inc.,
#   All Rights Reserved.
#
# PROPRIETARY - BELLCORE AND AUTHORIZED CLIENTS ONLY.
#
# This document contains proprietary information that shall
# be distributed or routed only within Telcordia Technologies
# (Telcordia), and its authorized clients, except
# with written permission of Telcordia.
#
# $Id: getHostLoc.pl,v 1.1 1999/05/20 22:27:07 rmartija Exp rmartija $
#

require 'getopts.pl' ;

undef;

$USAGE = "Usage: " . $0 . " [-D] -u file -m file\n" .
         "Flags:\n" .
         "   -D         debug mode\n" .
         "   -u file    file containing the list of unclassified IP\n" .
         "              addresses (i.e. those with unknown locations)\n" .
         "              and their characteristics.\n" .
         "   -m file    file containing the means and inverse of covariance\n" .
         "              matrices\n" .
         "Examples:\n" .
         "   $0 -u unknowns -m matrix" ;


%g_means = ();
%g_inverse = ();
@g_locales;
$g_debug;
$g_attributes;


#-----------------------------------------------------------------
#-----------------------------------------------------------------
sub getDistance {
    my( $loc, $data ) = @_;

    my( @X ) = @$data;

    my( @mu ) = @{$g_means{$loc}};
    my( @sigma ) = @{$g_inverse{$loc}};

    my( @diff, @prod );
    my( $i, $j );

    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $diff[$i] = $mu[$i] - $X[$i];
    }

    #
    # compute diff(transpose) * sigma. diff(transpose) is a 1 x N matrix
    # and sigma is a N x N matrix. the result is a 1 x N matrix.
    #
    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $prod[$i] = 0.;
        for( $j = 0; $j <= $g_attributes; $j++ ) {
            $prod[$i] += $diff[$j] * $sigma[$i][$j];
        }
    }

    #
    # multiply the matrix obtained above, i.e prod, with diff. prod is a
    # a 1 x N matrix and diff is a N x 1 matrix. the result is a scalar.
    #
    my( $dist ) = 0;
    for( $i = 0; $i <= $g_attributes; $i++ ) {
        $dist += $prod[$i] * $diff[$i];
    }

    return $dist;
}


#------------------------------------------------------------------
#------------------------------------------------------------------
sub readMeansAndMatrices {
    my( $file ) = @_;

    open( F, "< $file" );
    @lines = <F>;
    close( F );

    my( $n_rows, $cur_row, $line_num ) = (-1, 0, 0);
    my( $cur_loc, $n_means );

    foreach( @lines ) {
        chop;

        $line_num++;

        next if $_ =~ /^\s*$/;    # skip blank lines

        if( $_ =~ /^US.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $n_rows < 0;

            # $1 contains the state string (e.g. NJ)
            $cur_loc = "$1,US";
            $cur_row = 0;
        }
        elsif( $_ =~ /^NONUS.*:\s*(.*)/ ) {
            die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                unless $n_rows < 0;

            # $1 contains the country string (e.g. BE)
            $cur_loc = "$1,$1";
            $cur_row = 0;
        }
        elsif( $_ =~ /^MEAN.*:\s*(.*)/ ) {
                die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                    unless $n_rows < 0;

                # $1 contains something like 18.43 1130.71 20.00 170.71 19.57 228.5
                my( @means ) = split( ' ', $1 );
                $n_means = $n_rows = $#means;
                $g_means{$cur_loc} = \@means;
            }
            elsif( $_ =~ /^INVERSE.*:\s*(.*)/ ) {
                die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                    unless $cur_row == 0;
            }
            elsif( $_ =~ /^([A-Za-z]+).*:/ ) {
                die "ERROR: Invalid Tag in $file\n-> line $line_num: $_\n";
            }
            else {
                my( @row ) = split( ' ', $_ );

                # make sure the matrix is a $n_means X $n_means array
                die "ERROR: $file is corrupted\n-> line $line_num: $_\n"
                    unless $#row == $n_means && $cur_row <= $n_means;

                my( $r_entry ) = [@row];
                push( @{$g_inverse{$cur_loc}}, $r_entry );

                $cur_row++;
                $n_rows--;
            }
        }

        die "ERROR: $file is corrupted. More data expected.\n" unless $n_rows < 0;

        @g_locales = keys %g_means;
        return $n_means;
    }


    #-------------------------------------------------------------------
    #-------------------------------------------------------------------
    sub classifyIPs {
        my( $file ) = @_;

        open( F, "< $file" );

        my( @data, $tloc, $loc, %dist, $min );

        while( <F> ) {
            chop;
            next unless $_ =~ /^(\d+)\.(\d+)\.(\d+)\.(\d+).*\s*(.*)/;

            ($ip, @data) = split( ' ' );

            next unless $#data == $g_attributes;

            $min = time;    # initialize $dist to some arbitrary large number
                            # such as the number of seconds since 1/1/1970
            foreach $tloc ( @g_locales ) {
                $dist{$tloc} = &getDistance( $tloc, \@data );
                if( $dist{$tloc} < $min ) {
                    $min = $dist{$tloc};
                    $loc = $tloc;
                }
            }

            if( $g_debug ) {
                foreach $key (sort keys %dist) {
                    printf "%-15s   %-8s   %7.2f\n", $ip, $key, $dist{$key};
                }
            }

            printf "%-15s %-8s\n", $ip, $loc;
        }

        close( F );
    }


    #################################################################
    ####################### main program ###########################
    #################################################################

    $x = &Getopts( 'u:m:D' );
    die "$USAGE\n" unless ($x ne '');
    die "$USAGE\n" unless ($opt_u && $opt_m) ;

    die "ERROR: cannot open $opt_u\n" unless -e $opt_u;
    die "ERROR: cannot open $opt_m\n" unless -e $opt_m;

    $g_debug = 1 if( $opt_D );
    $g_attributes = &readMeansAndMatrices( $opt_m );
    &classifyIPs( $opt_u );
```